Software Requirements Specification (SRS) Document
Sprint Notes.docx

Active Members:

- Nathan Nguyen [s3948938]
- Jonathan Theofilas [s3942976]

Introduction

1.1 Overview

The SuperPrice application provides a one-stop solution for grocery shoppers, streamlining price comparisons, product searches, and delivery arrangements.

1.2 Purpose

To ensure users efficiently compare prices, arrange deliveries, and receive notifications on deals, enhancing their grocery shopping experience.

1.3 Stakeholders

- Project Developers
- Partnered Supermarkets and stores
- End-users (Shoppers)
- Client

1.4 System Scope

The SuperPrice application aims to cover a broad range of supermarkets for better product comparison, efficient delivery organization, and timely notifications for users.

Functional Requirements

Product Search and Categorization:

- Users can search specific products.
- Products are categorized for intuitive navigation.
- **Price Comparison**:
- Prices from diverse supermarkets for a product are displayed.
- The most economical price is emphasized.

Non-Functional Requirements

- **Performance**: Application loads within 1 seconds.
- **Security**: User data encryption is mandatory. Regular security checks recommended.
- **Usability**: The design is user-friendly for both beginners and adept users.
- **Reliability**: 99.5% system uptime.

- Scalability: Capable of handling increased user loads during peak shopping hours.
- Integrations: Seamless integration with supermarket APIs and third-party delivery services.

System Architecture

Need to insert SA Diagram. Show the relationships and data flow between components. [Member dropped out]

Data Model

Ensure the inclusion of the "Store" entity and the establishment of relationships among tables, including the foreign key "Store_ID". [Member dropped out]

Assumptions and Constraint

Assumptions:

- Supermarkets provide current prices.
- Users have internet connectivity.

Constraints:

- Application limited to collaborating supermarkets.
- Delivery areas restricted.

Dependencies

- **External APIs**: Connection with supermarket databases for real-time price details.
- **Libraries**: User-friendly interface UI/UX libraries and encryption libraries for data protection.

Testing and Acceptance Criteria

- **Functionality Testing**: Ensure all functionalities work as intended.
- **Security Testing**: User data is secure, and vulnerabilities are absent.
- **Usability Testing**: Users navigate effortlessly through the application.

Glossary

- **Supermarket**: Comprehensive store offering a range of food and household items.
- **API**: Set of protocols enabling software components to communicate.

Final SRS Notes:

- Opted to use H2 database for storing grocery items

Features, User Stories and Product Backlog

1. Filter by Supermarkets

User Story (US1.1): As a user, I want to filter products based on the supermarkets, so I can easily find products from my preferred store.

Estimation: 3 points

Priority: High

Test Case:

Navigate to the product list page.

Select a specific supermarket from a dropdown list or checkboxes.

View products exclusive to the selected supermarket.

DoD:

Accurate listing of products from the selected supermarket.

The filter can be easily cleared or changed.

SQL Query:

SELECT * FROM products WHERE supermarket = 'selected_supermarket';

User Story (US1.2): As a user, I want to compare the price of a specific product across different supermarkets so I can get the best deal.

Estimation: 4 points

Priority: High

Test Case:

Search for a specific product.

View the list of prices from various supermarkets for that product.

DoD:

Display all available supermarkets with the product's price.

Highlight or sort by the cheapest price for that product.

SQL Query:

SELECT supermarket, price FROM products WHERE name = 'product_name';

2. Search by Product Name

User Story (US2.1): As a user, I want to search for a product by its name, so I can quickly find what I am looking for.

Estimation: 3 points

Priority: High

Test Case:

Enter the product name in the search bar.

View the list of products matching the query.

DoD:

Display products that match the search query.

If no match is found, inform the user accordingly.

SQL Query:

SELECT * FROM products WHERE name LIKE '%search_query%';

User Story (US2.2): As a user, I want to see the most searched products, so I can be updated on trending products.

Estimation: 4 points

Priority: Medium

Test Case:

Navigate to the homepage or a dedicated 'Trending' section.

View the list of most searched products.

DoD:

Display a list of products based on their search frequency.

Regularly update this list to reflect recent trends.

SQL Query (requires additional infrastructure to track search frequency):

SELECT name, COUNT(search_frequency) AS frequency FROM products GROUP BY name ORDER BY frequency DESC LIMIT 10;

3. Add Products to Favorites

User Story (US3.1): As a user, I want to add products to my favorites, so I can quickly access them later.

Estimation: 3 points

Priority: Medium

Test Case:

Browse or search for a product.

Click on the 'Add to Favorites' or 'Heart' icon.

Access the 'Favorites' section to view the saved products.

DoD:

Successful addition of a product to favorites.

Easy access to the user's favorite products list.

SQL Query (requires a favorites table):

INSERT INTO favorites(user_id, product_id) VALUES ('user_id', 'product_id');

User Story (US3.2): As a user, I want to remove products from my favorites, so I can manage my list.

Estimation: 2 points

Priority: Low

Test Case:

Navigate to the 'Favorites' section.

Click on the 'Remove' or 'Unheart' icon beside the product.

Confirm that the product has been removed from the favorites list.

DoD:

Successful removal of a product from favorites.

The product no longer appears in the user's favorite products list.

SQL Query:

DELETE FROM favorites WHERE user_id = 'user_id' AND product_id = 'product_id';

These are some potential features and user stories you can implement in your SuperPrice website. Depending on your infrastructure, you might need to adjust the SQL queries accordingly.

4. Price Range Filter

User Story (US4.1): As a user, I want to filter products within a certain price range, so I can find items that fit within my budget.

Estimation: 4 points

Priority: High

Test Case:

Navigate to the product list page.

Set a minimum and maximum price using sliders or input fields.

View products within the selected price range.

DoD:

Only products within the specified price range are shown.

Filter can be easily adjusted or reset.

SQL Query:

SELECT * FROM products WHERE price BETWEEN 'min_price' AND 'max_price';

5. Sort by Price

User Story (US5.1): As a user, I want to sort the products by price, either ascending or descending, so I can easily identify the cheapest or the most expensive products.

Estimation: 3 points

Priority: Medium

Test Case:

Navigate to the product list page.

Use a dropdown or toggle button to select the desired sorting order.

View products sorted by their prices.

DoD:

Products are correctly sorted based on user's choice.

User can easily toggle between ascending and descending order.

SQL Query for ascending:

SELECT * FROM products ORDER BY price ASC;

For descending:

```
SELECT * FROM products ORDER BY price DESC;
```

6. Latest Products

User Story (US6.1): As a user, I want to see the latest products added to the database, so I'm always updated with new items.

Estimation: 4 points

Priority: Medium

Test Case:

Navigate to the homepage or dedicated 'New Arrivals' section.

View the list of the latest added products.

DoD:

Display a list of products based on their addition date.

Products are correctly tagged as 'New' or identified distinctly.

SQL Query (requires a timestamp or date column added_date in the table):

```
SELECT * FROM products ORDER BY added_date DESC LIMIT 10;
```